Допълнителни възможности на шел

ас. Стоян Мечев
катедра „Информационни технологии“
ВВМУ „Н. Й. Вапцаров“

Променливи на шел

- Шел ползва променливи, подобно на езиците за програмиране.
- Видими са само в шела, в който са били зададени.
- Имената се състоят само от букви, цифри и "_". Започват с буква или "_".
- Задаване на стойност на променлива
  - **var=1** (Без интервали!)
- Извеждане на стойност на променлива
  - **echo $var**
- Могат да бъдат експортирани към работната среда
  - **export var**
  - **export var1="My variable"** – едновременно деклариране и експортиране.
  - Наследяващ процес не може да експортира променлива към родителски процес.
- Изтриване на променлива:
  - **unset var** – изтрива променливата едновременно от шел и от средата.
  - **export -n var** – премахва я от средата, остава променлива на шел.

Променливи на работната среда (обкръжението)

- Всички променливи на работната среда са достъпни в шел.
  - Обратното не е валидно!
- Променливите на работната среда са достъпни за наследяващи процеси (не е задължително да е шел).
- Извеждане на променливите на средата

  echo $USER

  echo $?
- Извеждане на всички променливи на средата
  - **env** – без параметри
    - env – стартира процес, като му предава променлива
    - env [OPTION]... [-] [NAME=VALUE]... [COMMAND [ARG]...]
      - -u, --unset=NAME премахва променлива от средата
  - **export** – задаване на атрибута за изнасяне на променливите на обвивката.
    - export [-fn] [ИМЕ[=СТОЙНОСТ]...] или export -p
      - -f ИМЕ – функция на обвивката
      - -n Премахване на атрибута за изнасяне от всяко от ИМЕната
      - -p Извеждане на списък с имената на всички променливи на шел и функции за изнасяне
  - **set** – промяна на стойностите на позиционните параметри и опциите на шел, извеждане на имената и стойностите на променливите на шел.

Основни променливи на средата

Case sensitive

Name	Function
USER	The name of the logged-in user
UID	The numeric user id of the logged-in user
HOME	The user's home directory
PWD	The current working directory
SHELL	The name of the shell
PPID	The process id of the process that started this process (that is, the id of the parent process)
PATH	List of directories containing executable programs that are eligible as external commands
EDITOR	Name of the user's favourite editor
PS1	Shell command prompt template

PATH=$PATH:/test

Параметри, които не могат да бъдат променяни

Name	Function
$	The process id or PID of the running bash shell (or other) process
?	The exit code of the last command
0	The name of the shell or shell script

Псевдоними

Синтаксис

alias [-p] [име [= стойност] ...]

unalias [-a] [име ...]

Опции

-p Отпечатване на текущите стойности

-a Премахване на всички псевдоними

- Стойността не може да съдържа никакви позиционни параметри ($ 1 и т.н.), ако трябва да го направите, вместо това използвайте функция на шел.

- Името не може да бъде „alias“ или „unalias“.

  unalias може да се използва за премахване на всяко име от списъка с дефинирани псевдоними.

- Постоянни псевдоними
  - Използвайте любимия си текстов редактор, за да създадете файл, наречен ~/.bash_aliases, и въведете командите alias във файла.

Функции на шел

- Могат да се декларират директно в шел;
- Започват с ключова дума и ();
- Блокът се загражда с {};
- Приемат позиционни параметри.

```
testfunc () { echo "$#parameters"; echo "$@"; }
```

```
$ set|grep IFS
IFS=$' \t\n'
```

Параметри на функции и скриптове за шел

Параметър	Предназначение
0, 1, 2, …	Позиционни параметри, започвайки от параметър 0. Параметър 0 съответства на името на програмата или на името на скрипта, който използва функцията. Стринг затворен в единични или двойни кавички се приема като един параметър. Помнете за експандването на променливите, когато има двойни кавички.
*	Позиционните параметри, започвайки от параметър 1. Ако са използвани двойни кавички. Ако разширяването се извършва в двойни кавички, разширяването е една дума с първия символ на специалния разделител (interfield separator = IFS), разделяща параметрите, или без интервали, ако IFS е null. По подразбиране стойността на IFS е празна, табулация и нов ред. Ако IFS не е зададен, използваният разделител е празен, както за IFS по подразбиране.
@	Позиционните параметри, започвайки от параметър 1. Ако експандването е направено с двойни кавички, всеки параметър преминава като единична дума, така "$@" е равносилно на "$1", "$2", … Препоръчва се, ако параметрите съдържат вътрешни символи за интервал.
#	Броят параметри, без да се включва параметър 0.

Команди от файлове

- Изпълнение в субшел – не влияе на променливите на родителския шел
  - **bash** my_commands
    - не е необходимо файлът да е изпълним;
  - изпълним файл

    [#!/bin/bash]

    chmod u+x my_commands

    ./my_commands
- Изпълнение в текущия шел
  - source my_commands – чете командите, все едно, че ги въвеждате от клавиатурата. Работи в текущия шел.
  - . my_commands
- Търсят се в променливата PATH, както другите външни команди.
- командата **which** – намира команда
- командата **whereis** – намира команда, изходния код и ръководството към нея

Шел като език за програмиране

- Параметри
  - **<u>както при функциите.</u>**
- Цикли
  - for
    - използва списък от стойности разделени с интервал.
    - стойностите могат да се зададат в отделна променлива.
    - ако се пропусне ключовата дума "in", цикълът работи с параметрите подадени към скрипта.
  - Алтернативи "&&" и "||"

```
for i in 1 2 3
do
 echo And $i!
done
```

```
list='4 5 6'
for i in $list
do
 echo And $i!
done
```

```
for f
do
echo `wc -l <"$f"` lines in $f
done | sort -n
$ ./sort-wc /etc/passwd
/etc/fstab /etc/motd
```

```
for name
do
 test -d "$name" && echo $name: directory
 test -f "$name" && echo $name: file
 test -L "$name" && echo $name: symbolic link
done
```

Основни опции за командата test

- Логически изрази
  - test EXPRESSION1 -a EXPRESSION2 ⇔ and
  - test EXPRESSION1 -o EXPRESSION2 ⇔ or
- Стрингове
  - test -n STRING дължината на STRING не е нула
  - test STRING1 = STRING2 сравнява дали двата стринга са еквивалентни
  - STRING1 != STRING2 сравнява дали двата стринга не са еквивалентни
- Сравняване на цели числа
  - INTEGER1 -eq INTEGER2 ⇔ INTEGER1 = INTEGER2
  - INTEGER1 -ge INTEGER2 ⇔ INTEGER1 >= INTEGER2
  - INTEGER1 -gt INTEGER2 ⇔ INTEGER1 > INTEGER2
  - INTEGER1 -le INTEGER2 ⇔ INTEGER1 <= INTEGER2
  - INTEGER1 -lt INTEGER2 ⇔ INTEGER1 < INTEGER2
  - INTEGER1 -ne INTEGER2 ⇔ INTEGER1 != INTEGER2
- Опции за файлове
  - test -d FILE дали FILE съществува и е директория
  - test -e FILE дали FILE съществува
  - test -f FILE дали FILE съществува и е обикновен файл
  - test -L FILE дали FILE съществува и е линк
  - test -r FILE дали FILE съществува и може да бъде четен

Командата if

- Може да се използват [ ] вместо командата **test**
- Структура:

If <проверка на условие 1>
then
<команда/и>
[elif <проверка на условие 2>
then
<команда/и>
...
else]
fi

Тази част от командата може да бъде пропусната

```
for name
do
 if test -L "$name"
 then
  echo $name: symbolic link
 elif [ -d "$name" ]; then
  echo $name: directory
 elif [ -f "$name" ]
 then
  echo $name: file
 else
 echo $name: no idea
 fi
done
```

Използване на **[ ]** вместо **test** - обърнете внимание на интервалите

Когато **then** е на същия ред – използвайте **;** за разделяне на командите

Още цикли

- while
- until
- break
- continue

```
$ for i in `seq 1 3`
> do
> echo $i
> done
1
2
3
```

```
COUNTER=0
while [ $COUNTER -lt 10 ]
do
 echo The counter is $COUNTER
 let COUNTER=COUNTER+1
done
```

```
COUNTER=20
until [ $COUNTER -lt 10 ]
do
 echo COUNTER $COUNTER
 let COUNTER-=1
done
```

- **seq** - отпечатва номера от ПЪРВО до ПОСЛЕДНО, на стъпки от СТЪПКА

  seq [ОПЦИЯ]... ПОСЛЕДНО

  или: seq [ОПЦИЯ]... ПЪРВО ПОСЛЕДНО

  или: seq [ОПЦИЯ]... ПЪРВО СТЪПКА ПОСЛЕДНО

```
$ seq 0 0.1 0.3
0,0
0,1
0,2
0,3
```

- -f, --format=FORMAT използва форматиране в стил printf
- -s, --separator=STRING използва STRING като разделител. По подразбиране е "\n"
- Ако ПЪРВО или СТЪПКА са пропуснати, default е 1. Това е в сила, дори когато ПОСЛЕДНО е по-малко от ПЪРВО
- СТЪПКА и ПОСЛЕДНО се обработват като стойности с плаваща запетая.

```
$ seq -s"; " 0 0.1 0.3
0,0; 0,1; 0,2; 0,3
```

Командата read

```
$ read -p "Въведете две числа " a1 a2
Въведете две числа 1 2 3
$ echo $a1&&echo $a2
1
2 3
```

- **read** [-ers] [-a МАСИВ] [-d РАЗДЕЛИТЕЛ] [-i ТЕКСТ] [-n БРОЙ_ЗНАЦИ] [-N БРОЙ_ЗНАЦИ] [-p ПОДСКАЗКА] [-t БРОЙ_ЗНАЦИ] [-u ФАЙЛОВ_ДЕСКРИПТОР] [ИМЕ…]
- Изчитане на ред от стандартния вход и разделянето му по полета.

От стандартния вход или от файловия дескриптор ФД, ако е използвана опцията „-u“, се прочита един ред. Редът се разделя на полета — думи. Първата дума се присвоява на първото ИМЕ, втората дума на второто ИМЕ и т.н., а на последното ИМЕ се присвояват оставащите думи. Като разделители на думи се използват само знаците указани в променливата „IFS“.

Ако не са дадени ИМЕна, прочетеният ред се запазва в променливата „REPLY“.

Опции:

- -a прочетените думи се присвояват последователно на елементите на МАСИВа, като индексът му започва от 0.
- -d РАЗДЕЛИТЕЛ четенето продължава до прочитането на първия знак, който присъства в променливата „DELIM“, а не до минаването на нов ред.
- -n БРОЙ_ЗНАЦИ четенето завършва след прочитането на този БРОЙ_ЗНАЦИ, не се чака за нов ред. Разделител в рамките на този БРОЙ_ЗНАЦИ се зачита.
- -N БРОЙ_ЗНАЦИ четенето завършва с прочитането на точно този БРОЙ_ЗНАЦИ, освен ако не се появи EOF или времето за изчакване на въвеждане не изтече. Всички разделители се пренебрегват.
- -p ПОДСКАЗКА извежда низа ПОДСКАЗКА без минаване на нов ред, преди да започне четенето на знаци от входа.

Благодаря за вниманието!
Въпроси?